Encoding Partial Plans for Heuristic Search

Pascal Bercher and **Susanne Biundo**

Institute of Artificial Intelligence,
Ulm University, D-89069 Ulm, Germany,
firstName.lastName@uni-ulm.de

Abstract

We propose a technique that allows any planning system that searches in the space of partial plans to make use of heuristics from the literature which are based on search in the space of states.

The technique uses a problem encoding that reduces the problem of finding a heuristic value for a partial plan to finding a heuristic value for a state: It encodes a partial plan into a new planning problem, s.t. solutions for the new problem correspond to solutions reachable from the partial plan. Evaluating the goal distance of the partial plan then corresponds to evaluating the goal distance of the initial state in the new planning problem.

Introduction

In most of today's classical planning approaches, problems are solved by informed (heuristic) progression search in the space of states. One reason for the big success of this approach is the availability of highly informed heuristics performing a goal-distance estimate for a given state. In plan-space-based search, search nodes correspond to partially ordered partial plans. One of the most important representatives of this technique is partial-order causal link (POCL) planning (McAllester and Rosenblitt 1991; Penberthy and Weld 1992). The least commitment principle of POCL planning seems to be advantageous compared to the more restricted state-based search techniques, as it enforces decisions such as variable bindings, only if necessary. POCL planning has greater flexibility at plan execution time (Muise, McIlraith, and Beck 2011) and eases the integration for handling resource or temporal constraints and durative actions (Vidal and Geffner 2006; Coles et al. 2010). Its knowledge-rich plans furthermore enable the generation of formally sound plan explanations (Seegebarth et al. 2012). However, due to the complex structure of partial plans, developing well-informed heuristics for POCL planning is a challenging task (Weld 2011) and heuristics are still rare. To address the lack of informed heuristics for POCL planning, we propose an idea of how to use heuristics already known from state-based search, rather than developing new *specific* heuristics.

POCL Planning

A planning domain is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, where $\mathcal{V}$ is a finite set of state variables and $\mathcal{A}$ is a finite set of actions, each having the form (pre, add, del), where $pre, add, del \subseteq \mathcal{V}$. $2^{\mathcal{V}}$ is the set of states and an action is applicable in a state $s \in 2^{\mathcal{V}}$ if its precondition pre holds in s, i.e., $pre \subseteq s$. Its application generates the state $(s \setminus del) \cup add$. The applicability and application of action sequences is defined as usual. A planning problem in STRIPS notation is a tuple $\pi = \langle \mathcal{D}, s_{init}, g \rangle$ with $s_{init} \in 2^{\mathcal{V}}$ being the initial state and $g \subseteq \mathcal{V}$ being the goal description. A solution to π is an applicable action sequence starting in s_{init} and generating a state $s' \supseteq g$ that satisfies the goal condition.

POCL planning is a technique that solves planning problems via search in the space of partial plans. A *partial plan* is a tuple $(PS, \prec, CL)$. PS is a set of plan steps, each being a pair $l{:}a$ with an action $a \in \mathcal{A}$ and a unique label $l \in L$ with L being an infinite set of label symbols to differentiate multiple occurrences of the same action within a partial plan. The set $\prec \subset L \times L$ represents ordering constraints and induces a partial order on the plan steps in PS. CL is a set of causal links. A causal link $(l, v, l') \in L \times \mathcal{V} \times L$ testifies that the precondition $v \in \mathcal{V}$ of the plan step with label l' is provided by the action with label l. That is, if $l{:}(pre, add, del) \in PS$, $l'{:}(pre', add', del') \in PS$, and $(l, v, l') \in CL$, then $v \in add$ and $v \in pre'$. Furthermore, we demand $l \prec l'$ if $(l, v, l') \in CL$.

Now, π can be represented as a POCL planning problem $\langle \mathcal{D}, P_{init} \rangle$, where $P_{init} := (\{l_0{:}a_0, l_\infty{:}a_\infty\}, \{(l_0, l_\infty)\}, \emptyset)$ is the *initial partial plan*. The actions a_0 and a_∞ encode the initial state and goal description: a_0 has no precondition and s_{init} as add effect and a_∞ has g as precondition and no effects. A solution to a POCL planning problem is a partial plan P with no *flaws*. There are two flaw classes: $\mathcal{F}_{OpenPrecondition}$ and $\mathcal{F}_{CausalThreat}$. An *open precondition* in $\mathcal{F}_{OpenPrecondition}$ is a tuple $(v, l) \in \mathcal{V} \times L$ and specifies that the precondition v of the plan step with label l is not yet protected by a causal link. A *causal threat* in $\mathcal{F}_{CausalThreat}$ is a tuple $(l, (l', v, l'')) \in L \times CL$ and specifies that the ordering constraints $\prec$ allow the plan step $l{:}(pre, add, del)$ with $v \in del$ to be ordered in such a way that $\prec \cup \{(l', l), (l, l'')\}$ induces a partial order. That is, the plan step with label l *threatens* the causal link (l', v, l''), since it might undo its protected condition v.

If a partial plan P has no flaws, then every linearization of its plan steps that respects the ordering constraints is a solution to the corresponding planning problem π in STRIPS notation.

POCL planning can be regarded as a refinement procedure (Kambhampati 1997), since it *refines* the initial partial plan P_{init} step-wise until a solution is generated. To that end, first a partial plan P is selected, which is based on heuristics estimating the goal-distance or quality of P. Given such a partial plan P, a flaw selection function selects one of its flaws and resolves it. For that end, all *modifications* are generated, which are all possibilities to resolve the given flaw. There are three modification classes, each specifying modifications addressing certain flaw classes. A causal threat flaw $(l, (l', v, l'')) \in \mathcal{F}_{CausalThreat}$ can only be resolved by *promotion* or *demotion*. Promotion and demotion modifications belong to the class of $\mathcal{M}_{InsOrdering}$ and are ordering constraints, which promote the plan step with label l before the one with label l' or demote it behind the one with label l''. An open precondition flaw $(v, l) \in \mathcal{F}_{OpenPrecondition}$ can only be resolved by inserting a causal link (l', v, l) which protects the open precondition v. This can be done either by using a plan step already present in the current partial plan, or by a new action from $\mathcal{A}$ – the corresponding modification classes are $\mathcal{M}_{InsCausalLink}$ and $\mathcal{M}_{InsAction}$, respectively.

The procedure of selecting a partial plan, calculating its flaws, and selecting and resolving a flaw is repeated until a partial plan P without flaws is generated. Hence, P is a solution to the POCL planning problem and returned.

Heuristics for POCL Planning

In this section we briefly review the current state of the art heuristics for selecting a partial plan in POCL planning.

Although there are many heuristics for POCL planning, most of them are based on pure syntactical criteria like the number of open precondition flaws or the ratio of certain flaws to the number of plan steps, etc. (Younes and Simmons 2003; Schattenberg 2009). However, we are only aware of *two* heuristics for POCL planning which are based on a well-informed *means-ends analysis*: the *Additive Heuristic for POCL Planning* h^r_{add} (Younes and Simmons 2003) and the *Relax Heuristic* h_{relax} (Nguyen and Kambhampati 2001). The first one is a variant of the add heuristic (Haslum and Geffner 2000), whereas the second one can be regarded as a variant of the FF heuristic (Hoffmann and Nebel 2001).

While these heuristics are the currently best-informed heuristics available for (non-temporal) POCL planning, they both ignore the negative effects of the plan steps in the current partial plan, although those could be used to strengthen their heuristic estimates. In contrast to that, our technique allows, in principle, heuristics to use all information given by the current partial plan. To pinpoint our observation, we briefly review h^r_{add}[1].

The heuristic h^r_{add} takes as input a set of open preconditions of the partial plan and estimates the effort to achieve them based on a reachability analysis assuming sub-goal independence and delete relaxation.

Let $\langle\langle \mathcal{V}, \mathcal{A}\rangle, P_{init}\rangle$ be a POCL planning problem, $V \subseteq \mathcal{V}$ a set of state variables, $v \in \mathcal{V}$ such a state variable, $A(v) := \{(pre, add, del) \in \mathcal{A} \mid v \in add\}$ the set of actions with an add effect v, and $a := (pre, add, del) \in \mathcal{A}$ an action. Then, h^r_{add} is based on the following functions:

$$h^{variables}_{add}(V) := \sum_{v' \in V} h^{aVariable}_{add}(v')$$

$$h^{aVariable}_{add}(v) := \begin{cases} 0 & \text{if } v \in s_{init} \\ \min\limits_{a \in A(v)} h^{anAction}_{add}(a) & \text{if } A(v) \neq \emptyset \\ \infty & \text{else} \end{cases}$$

$$h^{anAction}_{add}(a) := 1 + h^{variables}_{add}(pre)$$

The heuristic $h^r_{add}(P)$, which does *reuse* actions in the current partial plan $P = (PS, \prec, CL)$, is now defined by $h^{variables}_{add}(g_P)$, where g_P is a subset of all open preconditions. Let $OC \subseteq \mathcal{V} \times L$ be the set of all open preconditions of P. Then, $g_P := \{v \mid (v, l) \in OC$ and there is no l':$(pre, add, del) \in PS$, s.t. $v \in add$ and the set $\prec \cup \{(l', l)\}$ induces a partial order$\}$. Thus, g_P is the set of all open preconditions for which new plan steps must be inserted in order to resolve these flaws.

It is easy to see that the given partial plan P and its structure are only used to identify open preconditions. Positive interactions are used only to a certain extent and negative interactions are completely ignored. Of course, the very idea of this heuristic is the delete relaxation; however, to ignore the negative effects of actions in PS is an *additional* relaxation, which might lead to a strong underestimation of the heuristic. The original version of the add heuristic for state-based search takes a current *state* s as input, and the heuristic assumes that all state variables of s remain true. Since there is no such state in our setting, h^r_{add} assumes that all state variables of s_{init} remain true. However, this assumption is much more severe than in the original version of the add heuristic, since in state-based search, s reflects *all effects* of *all actions* leading to s, whereas h^r_{add} does only incorporate the *positive effects* of all actions leading to P, but not its negative ones.

We argue that the plan structure and the positive *and negative* interactions of the plan steps given in the partial plan should be used to improve heuristic estimates. Our proposed technique allows to take all these factors into account.

New Heuristics for POCL Planning

Our idea to make the heuristics from state-based planning available to POCL planning involves encoding the current partial plan by means of an altered planning problem, s.t. estimating the goal distance for that *partial plan* corresponds to estimating the goal distance for the initial *state* in the new planning problem.

Please note that a similar encoding was already proposed by Ramírez and Geffner (Ramírez and Geffner 2009). However, their transformation was used in the context of plan recognition for compiling observations away.

[1] We do not review both heuristics, because h_{relax} is basically just an improvement of the add heuristic taking into account positive interactions to a larger extent.

Transformation

Our transformation works as follows: given a planning problem in STRIPS notation $\pi = \langle\langle \mathcal{V}, \mathcal{A}\rangle, s_{init}, g\rangle$ and a partial plan $P = (PS, \prec, CL)$, let $enc_P(\pi, P) = \langle\langle \mathcal{V}', \mathcal{A}'\rangle, s'_{init}, g'\rangle$ be the *encoding* of π and P with:

$$\begin{aligned}
\mathcal{V}' &:= \mathcal{V} \cup \{l_-, l_+ \mid l{:}a \in PS, l \notin \{l_0, l_\infty\}\} \\
\mathcal{A}' &:= \mathcal{A} \cup \{enc_{PS}(l{:}a, \prec) \mid l{:}a \in PS, l \notin \{l_0, l_\infty\}\}, \\
&\text{with } enc_{PS}(l{:}(pre, add, del), \prec) := \\
&\quad (pre \cup \{l_-\} \cup \{l'_+ \mid l' \prec l, l' \neq l_0\}, \\
&\quad\ add \cup \{l_+\}, del \cup \{l_-\}), \\
s'_{init} &:= s_{init} \cup \{l_- \mid l{:}a \in PS, l \notin \{l_0, l_\infty\}\} \\
g' &:= g \cup \{l_+ \mid l{:}a \in PS, l \notin \{l_0, l_\infty\}\}
\end{aligned}$$

The transformed problem subsumes the original one and extends it in the following way: all plan steps present in P are additional actions in $\mathcal{A}'$ – we do not encode the artificial start and end actions, since their purpose is already reflected by the initial state and goal description. The new actions use the labels of their corresponding plan steps as additional state variables to encode whether they have already been executed or not. Thus, for every label we introduce two new state variables: l_- for encoding that the corresponding plan step/action has not yet been executed and l_+ to encode that it has been executed. Initially, none of these plan steps were executed and the (additional) goal is to execute all of them. Furthermore, the new actions use these labels to ensure that they can only be executed in an order consistent with the ordering present in the plan to encode. Please note that we do not encode the causal links for the sake of simplicity, although it is possible.

Before we can state the central property of the transformed problem, we need some further definitions: $ref(P) := \{\langle PS', \prec', CL'\rangle \mid PS' \supseteq PS, \prec' \supseteq \prec, CL' \supseteq CL\}$ is called the set of all *refinements* of P, i.e., the set of all partial plans which can be derived from P by adding plan elements. Let $sol(\pi)$ be the set of all *solution* plans of π. Then, $sol(\pi, P) := sol(\pi) \cap ref(P)$ is the set of all solutions of π, which are refinements of P. The cost of P is denoted by $c(P) := |PS|$.

Theorem 1. *Let π be a planning problem and P a partial plan with no causal links. Then,*

$$\min_{P' \in sol(\pi, P)} c(P') = \min_{P' \in sol(enc_P(\pi, P))} c(P')$$

This theorem states that an optimal solution for π, which also has to be a refinement of P, has the same cost as an optimal solution for the transformed problem. To prove that theorem, we provide two propositions from which it directly follows. The first proposition states that every solution of the original planning problem, which is also a refinement of the given partial plan, does also exist as a solution for the encoded problem. The second proposition states that every solution of the encoded problem can be decoded into a solution of the original one, which is a refinement of the given partial plan, too. Before we can state these propositions formally, we have to show how partial plans derived from $enc_P(\pi, P)$ can be transformed back into plans for π.

Let the *decoding* of a plan step be given by $dec_{PS}(l{:}(pre, add, del)) := l{:}(pre \cap \mathcal{V}, add \cap \mathcal{V}, del \cap \mathcal{V})$ and the decoding of a partial plan be given by $dec_P(\langle PS, \prec, CL\rangle) := \langle\{dec_{PS}(l{:}a) \mid l{:}a \in PS\}, \prec, \{(l, v, l') \in CL \mid v \in \mathcal{V}\}\rangle$.

Proposition 1. *Let π be a planning problem, P a partial plan with no causal links, and $P_{sol} \in sol(\pi, P)$. Then, there exists a plan P'_{sol} with $P'_{sol} \in sol(enc_P(\pi, P))$ and $dec_P(P'_{sol}) = P_{sol}$.*

Proposition 2. *Let π be a planning problem, P a partial plan with no causal links, and $P'_{sol} \in sol(enc_P(\pi, P))$. Then, $dec_P(P'_{sol}) \in sol(\pi, P)$.*

Proof Sketch. Follows from construction. □

Theorem 1 can be exploited by using heuristics known from state-based planning in the context of POCL planning: we want to find a heuristic function $h(\pi, P)$ that estimates the goal distance in π from the *partial plan* P. To that end, we transform π and P into the planning problem $\pi' = enc_P(\pi, P)$ and set $h(\pi, P) := \max\{h_{sb}(\pi', s'_{init}) - c(P), 0\}$, where h_{sb} is any heuristic that takes a *state* as input. We subtract the action cost of P from the estimate, since the heuristic h has to estimate the distance from P, whereas h_{sb} estimates the goal distance from the new initial state thereby including the costs of the plan steps in P. We maximize with zero, in case the heuristic h_{sb} underestimates the optimal goal distance by returning a value smaller than the action costs of the given plan.

From Theorem 1, we can also conclude that we inherit admissibility: if h_{sb} is admissible, h is admissible, too. This is an important property of our technique, as the currently best-informed heuristics for POCL planning, h^r_{add} and h_{relax}, are both not admissible. Our technique thus provides POCL planning with the first admissible heuristics by using admissible heuristics from state-based planning.

Since our transformation ignores causal links, the encoded planning problem is already relaxed if the given partial plan has causal links. Thus, given a partial plan *with* causal links, the equality in Theorem 1 is weakened in such a way that the optimal solution cost of $sol(enc_P(\pi, P))$ is just a lower bound of the optimal cost of $sol(\pi, P)$.

Evaluation

In this section, we evaluate the overhead of performing the encoding process. We begin by examining the canonical approach where the transformation is done for each partial plan independently, followed by an analysis of the costs if one performs an *incremental* transformation. To that end, let $\pi := (\mathcal{V}, \mathcal{A}, s_{init}, g)$, $P := (PS, \prec, CL)$ be a partial plan, and $enc_P(\pi, P) = (\mathcal{V}', \mathcal{A}', s'_{init}, g')$.

We can directly observe that the elements of $enc_P(\pi, P)$ are supersets of the elements in π. The number of their additional elements (state variables and actions) is bounded by a constant factor in the number of plan steps in PS. However, the preconditions of the additional actions in $\mathcal{A}' \setminus \mathcal{A}$ need some further attention. The size of the subset $\{l'_+ \mid l' \prec l, l' \neq l_0\}$ of the precondition of such an action can be linear in the number of plan steps of P. We can hence conclude

that the transformation has a time and space consumption of $\Theta(|\pi| + |PS| + |\prec|)$, which is in $O(|\pi| + |PS|^2)$.

However, we want to minimize the overhead we incur by performing the transformation; thus, we desire an *incremental* encoding of the current partial plan, where the transformed planning problem depends only on the previous one and the modification applied last.

Theorem 2. *Let π be a planning problem, P a partial plan, m a modification resolving some flaw of P thereby generating P', and $\pi' := enc_P(\pi, P)$. Then, π' can be transformed into $\pi'' := enc_P(\pi, P')$ in $O(1)$, given P, and m.*

Proof. We give a constructive proof by providing an algorithm calculating $enc_{P\text{-}inc}(\pi', P, m) := (\mathcal{V}'', \mathcal{A}'', s''_{init}, g'')$, s.t. $enc_{P\text{-}inc}(\pi', P, m) = enc_P(\pi, P')$. The modification m can only belong to one of the classes $\mathcal{M}_{InsOrdering}$, $\mathcal{M}_{InsAction}$, and $\mathcal{M}_{InsCausalLink}$. If m is the insertion of an ordering constraint or a causal link, $\mathcal{V}''$, s''_{init}, and g'' are not changed w.r.t. the elements of $\pi' = (\mathcal{V}', \mathcal{A}', s'_{init}, g')$. If m is a task insertion, these sets are extended by just one or two elements, each. Their incremental construction can thus be performed in constant time. The more interesting part is the calculation of $\mathcal{A}''$. Let $m = (l, l') \in \mathcal{M}_{InsOrdering}$ and $enc_{PS}(l'{:}a, \prec) = (pre, add, del)$ be the encoding of the plan step in PS with label l'. This action must be altered in order to represent the new ordering constraint. Thus, $\mathcal{A}'' := (\mathcal{A}' \setminus \{(pre, add, del)\}) \cup \{(pre \cup \{l_+\}, add, del)\}$. Since we only remove and add one element, we can compute this set in constant time, assuming the set operations are constant-time bounded. Since the insertion of a causal link is only reflected via an ordering constraint, we obtain the same result for $m \in \mathcal{M}_{InsCausalLink}$. For $m \in \mathcal{M}_{InsAction}$, we also have to do the same as for the previous modification classes, as m inserts a new action $a \in \mathcal{A}$ and a causal link (l, v, l') from $l{:}a$ to $l'{:}a'$. In addition, we must insert the action $enc_{PS}(l{:}a, \emptyset)$, which can also be done in constant time. Please note that we can use an empty set of ordering constraints, since this set only determines which plan steps must precede $l{:}a$ - however, since $l{:}a$ is just being inserted, there are no such plan steps, yet. Hence, no further alterations must be made. We have thus shown that $enc_{P\text{-}inc}(\pi', P, m)$ can be calculated in constant time.

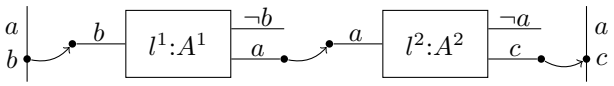
We do not show $enc_{P\text{-}inc}(\pi', P, m) = enc_P(\pi, P')$, since the proof is straight-forward. □

Given a partial plan P', its parent P and the encoding of P, $\pi' := enc_P(\pi, P)$, the theorem states that one can calculate the encoding of P', $\pi'' := enc_P(\pi, P')$, in constant time. However, note that the proof relies on an algorithm which *directly manipulates* π' in order to calculate π''. Thus, in case a partial plan has more than one successor, applying the algorithm given in the proof violates the premise that we have stored the encoding for every plan. To address that problem, it suffices to maintain a copy for every encoded planning problem, which can be done in linear time in the size of the given plan. (Since every encoding contains the original planning problem, it does not need to be copied for each individual partial plan.)

Please note that we only discussed the *runtime* of the *encoding process*. However, for our purpose of using the technique for calculating heuristic estimates, the *size* of the resulting problems is of more importance as the runtime of the heuristic calculation heavily depends on this size.

Example

Let $\pi = \langle\langle \mathcal{V}, \mathcal{A}\rangle, s_{init}, g\rangle$ be a planning problem with $\mathcal{V} := \{a, b, c\}$, $\mathcal{A} := \{(\{b\}, \{a\}, \{b\}), (\{a\}, \{c\}, \{a\})\}$, $s_{init} := \{a, b\}$, and $g := \{a, c\}$. Let P be a partial plan which was obtained by a POCL algorithm as depicted below:



The arrows indicate causal links and A^1 and A^2 are the two actions of $\mathcal{A}$. P has only one open precondition: (a, l_∞), which encodes the last remaining goal condition. Since a is already true in the initial state, both the add and the relax heuristic estimate the effort to be 0. However, the optimal goal distance is even ∞, since there is no refinement of P, which is a solution.

Due to Theorem 1, a heuristic based on the transformed problem can incorporate the negative effects of $l^1{:}A^1$ and $l^2{:}A^2$ and has thus the potential to discover the partial plan/state to be invalid and thus prune the search space. With $\mathcal{A}'$ being defined below, $enc_P(\pi, P)$ is given by $\langle\langle\{a, b, c, l^1_+, l^1_-, l^2_+, l^2_-\}, \mathcal{A}'\rangle, \{a, b, l^1_-, l^2_-\}, \{a, c, l^1_+, l^2_+\}\rangle$.

$$\begin{aligned}\mathcal{A}' := \{&(\{b\}, \{a\}, \{b\}\}),\\ &(\{b, l^1_-\}, \{a, l^1_+\}, \{b, l^1_-\})\\ &(\{a\}, \{c\}, \{a\}),\\ &(\{a, l^2_-, l^1_+\}, \{c, l^2_+\}, \{a, l^2_-\})\}\end{aligned}$$

Discussion

Relaxation *Every* (practically relevant) heuristic performs some kind of relaxation. Therefore, one must investigate which impact the relaxation of actions in $\mathcal{A}_{new} := \mathcal{A}' \setminus \mathcal{A}$ has for the resulting heuristic estimates. Since these actions encode the current partial plan, relaxing them would contradict the goal to use *all* information of the current planning progress. Thus, only relaxing the actions in $\mathcal{A}$, but none in $\mathcal{A}_{new}$ would improve the heuristic accuracy. However, one has to investigate how this can be done for each individual heuristic and how much it would influence the time to calculate its heuristic estimate. But, of course, relaxing them to a certain extent still captures *some* information obtained by the current partial plan.

Preprocessing Some heuristics, like merge and shrink abstraction (Dräger, Finkbeiner, and Podelski 2006; Helmert, Haslum, and Hoffmann 2007), perform a preprocessing step before the actual search and make up for it when retrieving each single heuristic value. Since we obtain a new planning problem for each single partial plan using the results of that preprocessing step might not be possible, directly. Thus, one would have to find a way of using this kind of heuristics in our setting, for instance by updating the result

of the preprocessing incrementally, as it can be done for the transformation itself.

Runtime Although we proved that the transformation itself can be done efficiently, we expect that the computational time of the used heuristics increases with the size of the partial plan to encode. This seems to be a strange property, since one would expect the heuristic calculation time either to remain constant (as for abstraction heuristics) or to *decrease* (as for the FF or add heuristics) as closer a partial plan comes to a solution. However, that might be a direct consequence from partial plans being complex structures, as many interesting decision problems involving them are NP hard w.r.t. their size (Nebel and Bäckström 1994).

Conclusion

We presented a technique which allows planners performing search in the space of plans to use standard classical planning heuristics known from state-based search. This technique is based on a transformation which encodes a given partial plan by means of an altered planning problem, s.t. evaluating the goal distance for the given partial plan corresponds to evaluating the goal distance for the initial state of the new planning problem. We proved that performing the transformation can be done incrementally in constant time under certain assumptions.

We conclude that our technique allows to fuse the benefits of the least-commitment principle and regression-like search of POCL planning with very strong heuristics known from state-based progression search.

An empirical evaluation showing the practical impact of using state-based heuristics in POCL planning is currently ongoing work.

Acknowledgements

This work is done within the Transregional Collaborative Research Centre SFB/TRR 62 “Companion-Technology for Cognitive Technical Systems” funded by the German Research Foundation (DFG).

References

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 42–49. AAAI Press.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed model checking with distance-preserving abstractions. In Valmari, A., ed., *SPIN*, volume 3925 of *Lecture Notes in Computer Science*, 19–34. Springer.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, 140–149. AAAI Press.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 176–183.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14:253–302.

Kambhampati, S. 1997. Refinement planning as a unifying framework for plan synthesis. *AI Magazine* 18(2):67–98.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI 1991)*, 634–639. AAAI Press.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2011. Monitoring the execution of partial-order plans via regression. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1975–1982. AAAI Press.

Nebel, B., and Bäckström, C. 1994. On the computational complexity of temporal projection, planning, and plan validation. *Artificial Intelligence* 66(1):125–160.

Nguyen, X., and Kambhampati, S. 2001. Reviving partial order planning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, 459–466. Morgan Kaufmann.

Penberthy, J. S., and Weld, D. S. 1992. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the third International Conference on Knowledge Representation and Reasoning*, 103–114. Morgan Kaufmann.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1778–1783. AAAI Press.

Schattenberg, B. 2009. *Hybrid Planning & Scheduling*. Ph.D. Dissertation, University of Ulm, Germany.

Seegebarth, B.; Müller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users – a formal approach for generating sound explanations. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 225–233. AAAI Press.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence* 170(3):298–335.

Weld, D. S. 2011. Systematic nonlinear planning: A commentary. *AI Magazine* 32(1):101–103.

Younes, H. L. S., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research (JAIR)* 20:405–430.